A WEIGHTED ROUND-ROBIN ARBITRATOR

FIELD OF THE INVENTION

[0001] This application is related to U.S. application Sr. No. _____ (TI-36014) filed on even date and, entitled "PCI Express Switch", which is incorporated herein by reference.

[0002] This invention relates to a weighted round-robin arbitrator and more specifically to a weighted round-robin arbitrator for a PCI-Express fabric.

BACKGROUND OF THE INVENTION

[0003] Peripheral Component Interconnect (PCI) is a parallel bus architecture developed in 1992 which has become the predominant local bus for personal computers and similar platforms. The implementation of this technology has come close to its practical limits of performance and can not easily be scaled up in frequency or down in voltage. A new architecture utilizing point-to-point transmission, having a higher speed, and which is scalable for future improvements, is knows as PCI Express.

[0004] The preferred technique for port arbitration between virtual channels (VC) in PCI Express is a weighted round-robin arbitrator. The advantage of the weighted round-robin arbitrator over a non-weighted round-robin arbitrator is that it is still possible to provide priority to one or more of the channels while guaranteeing that the channel having the least priority will still have access to the network. It is common in a weighted round-robin network to utilize a table having a predetermined number of time slots which are then allocated to the various channels that are vying for access to the

network. PCI Express, for example, typically used between 32 and 256 time slots. Priority is given to a particular channel by providing it with more of the time slots than other channels, for example, the highest priority channel might receive half of the time slots whereas the remainder of the time slots are divided amongst lower priority channels. Each channel must receive at least a single time slot so that it is guaranteed access to the network. The assigned time slots can be contiguous or non-contiguous, depending upon the needs of a particular system, and this is usually assigned via a software routine.

[0005] Figure 1 illustrates a known weighted round-robin arbitrator generally shown as 100. In this technique, after a packet of data has been sent out, a signal on line 102 advances the 3 bit counter 104 which address a look-up table 120. A 3 bit counter can address up to 8 entries in the look-up table. The output of the look-up table is a queue select signal on line 122 which is input into multiplexers 108, 124 and 128. Multiplexer 108 receives 3 empty flags 114, 116, 118 from queues 0, 1 and 2 respectively. The queue select signal on line 122 selects the queue matching the entry in the look-up table 120. If the selected queue has data to be sent out, the empty flag will show that that particular queue is not empty. The signal POP on line 130 will be multiplexed via multiplexer 128 onto the line 132, 134 or 136, depending on whether queue 142, 140 or 138 respectively, has been selected. On each POP signal, a bit will be transmitted from the queue via multiplexer 124 to the queue output data line 126. If there are 16 bits to a packet, for example, then 16 POP signals will be utilized to produce the 16 bits that is to be transmitted as a packet. If the selected flag indicates that the particular queue which has been selected is empty, the 3 bit counter 104 will continue to be advanced by the signal 102 to generate a different output to access the next entry in look-up table 120. This will continue until a queue having data to be sent out is found. If all the flags 114, 116, 118 show that all queues are empty AND gate 110 will generate a empty signal on line 112 to indicate to the system controller that there is

no data to be sent out. Data is loaded into the queues 138, 140, 142 via lines 144, 146 and 148 respectively.

[0006] The system shown in Figure 1 works quite well when there is data in the queue ready to be sent out. Unfortunately, in real life, often a queue will not have data that is ready to be sent out. In the system shown in Figure 1, it takes the entire clock cycle to reach this determination. Thus, if no data is ready to be sent out from the selected queue, the time slot is lost. Each time the signal 102 changes only a single queue will be assessed for data to be sent out. If the same queue number is utilized in a series of contiguous time slots or the queues assigned to a series of contiguous time slots all contain no data to be sent out, then it will take a considerable amount of time to reach a time slot assigned to a queue that does have data to be sent out, which time is irretrievably lost. For example if channel Ø (queue Ø) has data to be sent out but it is the lowest priority channel having only a single time slot in a 256 slot table, and all other queues do not have data to be sent out, then data will only be sent out once in 256 transitions of the signal 102. This means that 255 of the 256 time slots are wasted and the data from channel Ø will be unnecessarily delayed. Accordingly, there is a need for a weighted round-robin arbitrator that will not waste the time slot when the selected queue is empty.

SUMMARY OF THE INVENTION

[0007] It is a general object of the invention to provide a weighted round-robin arbitrator that can select the next highest priority queue that has data to be sent out.

[0008] This and other objects and features are provided, in accordance with one aspect of the invention by a weighted round-robin arbitrator comprising a plurality of data queues. An arbitration table comprises a plurality of entries, each entry representing a time slot for the transmission of one data packet from a selected one of

the plurality of data queues. An arbitration logic circuit, there being one such circuit for each of the plurality of entries in the arbitration circuit, comprises a first multiplexer receiving an output from a first table entry and an output from a second table entry in the arbitration table. A second multiplexer receives empty flags from each of the data queues, the flags indicating that there is no data to the sent from that queue, an output of the second multiplexer being coupled to a control input of the first multiplexer whereby the first table entry value is output from the first multiplexer if the corresponding queue has data to be sent out and the second table entry value is sent out from the first multiplexer if the queue corresponding to that table entry has data to be sent out and the queue corresponding to the first entry has no data to be sent out.

[0009] Another aspect of the invention includes a method of weighted round-robin arbitration comprising determining if a data queue corresponding to a first table entry in an arbitration table has data to be sent out. Sending out the data in the first entry if data to be sent out is present. If no data to be sent out is present in the data queue corresponding to the first entry, determined if data in a data queue corresponding to a second entry is ready to be sent out. Sending out the data in the second entry if data to be sent out is present.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Figure 1 is a block diagram of a know weighted round-robin arbitrator;

Figure 2 is a block diagram illustrating the concept of the present invention;

Figure 3 is a block diagram of a weighted round-robin arbitrator in accordance with the present invention; and

Figure 4 is a block diagram of the system of Figure 3 modified for use with a long table.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

[0011] Referring to Figure 2, a block diagram of a system according to the present invention is generally shown 200. The block diagram 200 has a slot entry table 202 showing eight slots or arbitration slices 204-218 in order to be comparable to the circuit shown Figure 1. It should be noted, however, that in PCI Express fabric, 32 to 256 slots would actually be used. Each of the slots 204-218 will store the address of one of the channels that is being arbitrated which will be selected by the system 200. The blocks do not store the data to be sent out, only the address of the queue holding the data. The number of bits stored by each slot 204-218 is determined by the number of channels in the system, where each channel has its separate queue for data to be sent out on that channel. The number of queues is not related to the number of slots in the table. The number of queues equals the number of channels to be arbitrated. The number of slots determines the resolution of the system. For example, with only 8 slots in the table 202, the time resolution is 12.5%. In contrast, a table having 256 slots has a resolution of $\frac{1}{256}$ or about 0.4%. If one desired to assign 10% of the time slots to channel Ø, for example, an 8 slot table is inadequate, but there is no problem with a 256 slot table. Each of the blocks 204-218 receives the empty flags from the respective queues on line 224. As shown here, three empty flags are coupled to the slots, which means there are three queues in the system. The highest priority address of the highest priority channel or queue is stored in slot 204. After a successful transmission from that channel, the table 202 will be shifted via line 220 so that the next highest priority channel, the address of which is stored in slot 206, will move to block 204 and become the new highest priority channel. As described above, it is possible that the address of the queue to be selected in both slots 204 and 206 are the same, so that particular channel is weighted heavier than other channels, for example.

[0012] However, in the system shown in Figure 2 generally as 200, if the flag on line 224 corresponding to the address of the queue stored in slot 204 is the same, then when the channel is to be selected for data transmission, the address stored in slot 204 will be bypassed and the next address, stored in slot 206 will be utilized. Likewise, if the flag indicates that the queue which is addressed by the address stored in slot 206 is empty, the next available queue, that is the one whose address is stored in slot 208 will be utilized. In this system, no time is lost in selecting for each time slot which address (queue) has data to be sent out because the bypassing of a particular entry in the table 202 is accomplished by means of combinational logic, and thus the bypass of the empty queues can be accomplished within a single clock cycle. Once data has been sent out, even if the data corresponds to the address stored in slot 210, the table will be rotated by a single position, so that the address stored in slot 206 will now be the first slot (slot 204) that has the highest priority. In another embodiment, once the data has been sent out, the slot from which the address was sent out will be rotated to the end of the table. All slots from that slot to the end of the table will then be rotated one slot to the left in the table. This preserves the relative priority of the empty slots. That is, if a slot that was bypassed is no longer empty, it will be considered for the next time slot.

[0013] Figure 3 shows a detailed implementation of the invention illustrated in Figure 2 generally as 300. The circuit 300 illustrates the circuitry for two contiguous blocks of the table, each representing a specific time slot. One such block is required for each slot in the table. The circuits 302, 352 are identical in construction and are designed to be modular so that the table can be configured from the number of such circuits required to make a table of the desired length. That is, for PCI Express implementation, typically 32 to 256 slots would be made available in the table. Therefore, 32 to 256 of the circuits 302 or 352 are required.

[0014] Each block has a table entry TE 308, 358 which contains the address of the queue to be selected when that slot is active. Each table entry 308, 358 receives a

POP signal on lines 304, 354 and a clock signal on lines 306, 356. Each block 302, 352 contains a multiplexer 312, 362 which receives the output of the table entry, and a daisy-chain input from the preceding block. The output of the multiplexer 312 on line 314, 364 is fed to the next stage. If the block, such as 302, is the first stage in the chain, the output is fed to the multiplexer that selects the queue to send out the data. The daisy-chain output of the last block in the chain (such as block 352) has its daisy-chain input tied to an arbitrary logic level (the diagram show a loop back from the first block). Each of the multiplexers 312, 362 receives the output of a multiplexer 316, 366. Empty flags 318, 320, 322 or 368, 370, 372 from the queues of the respective channels are input to the multiplexers 316, 366 respectively. The control input of the multiplexer 316, 366 is coupled to the TE outputs 310, 360 respectively. The TE output first stage 302 is fed back to the equivalent of 374 of the highest number stage (not shown) as shown by line 220 in Figure 2.

[0015] In operation, the output signal TE on line 310 is used to select one of the empty flags 318, 320, 322 input to multiplexer 316. If that empty flag indicates that the queue selected by the table entry on line 310 is empty, the signal 315 applied to multiplexer 312 will select the daisy-chain input and the queue corresponding to the table entry 310 is bypassed. If the empty flag applied to multiplexer 312 indicates that data is ready to be sent out from that queue the signal 315 applied to multiplexer 312 will select the queue number on line 310 which will be output on line 314 and applied to the queue multiplexer (not shown in Figure 3) to select the appropriate queue for data to be sent out.

[0016] If the daisy-chain input of multiplexer 312 has been selected, then the output of multiplexer 362 on line 364 will be chosen. The output of that multiplexer can be the TE value on line 360 or, if that value corresponds to a queue which has no data to be sent out, it will be the daisy-chain input 376 to multiplexer 362. In this matter, each stage which addresses a queue having no data to be sent out will be bypassed, all

within a single clock cycle. Thus, the slot will automatically go to the first queue which has data to be sent out. Once data has successful been sent out from the queue, the table is shifted so that the TE value in block 358 is transferred to block 308 and all the other blocks are transferred to the block to the left (not shown). On the next cycle priority will be give to the data which is now contained in table entry TE 308.

[0017] Thus, if there were 256 slots utilized and thus 256 entries into the table 202, and only a single block, the lowest priority block having been assigned a single slot 255, the system would bypass all of the preceding blocks which refer to queues which have no data to be sent out, and choose the block that does have data to be sent out, thus rapidly speeding up the transmission of this data. A technique for avoiding long time delays which could be generated by such a long table is discussed below in connection with Figure 4.

[0018] In a long table, the ripple effect of going through each multiplexer of each stage such as stages 302 and 352 could be excessive. In order to avoid this problem a circuit layout shown generally 400 is utilized. In this embodiment, the table is broken into two segments 402A-402M and 452A-452N, although any number of segments can be utilized. The number of slots within the table is represented by the numbers A-M and A-N where M or N can be equal to any integer and the number of segments in the first section of table 402A-402M may not be equal to the number of segments in the table sections 452A-452N. Each of the segments 402A-402M generates a signal 415 which is equivalent to the signal 315 generated in block 302 of Figure 3. These are the enable signals for the multiplexer 312 shown in Figure 3. If all the signals indicate that all the multiplexers in the blocks 402A-402M are to be bypassed, then there is no need for the ripple of all the multiplexers to occur. The signals 415A-415M are applied as inputs AND gate 412 having an output on line 410 which is coupled to the control input of multiplexer 404. The output from the first block 402A is coupled via line 406 to the multiplexer 404. In addition, the output from the first block 452A of the second segment

is coupled via line 408 to the second input of multiplexer 404. If all of the signals 415A-415M indicate that all the blocks 402A-442M are to be bypassed, then the multiplexer 404 will be activated to select the input 408 rather than the input 406 and thus reduce the time to reach an active slot. This circuitry is similar to the carry-look-ahead circuitry for an adder.

[0019] One major advantage of the present invention is that it is modular. Thus, the circuit required to generate the queue addresses requires the same number of identical blocks as the number of slots. The modular block 302, 252 or 402, 452 do not require circuit changes with changes in the length of the slot entry table. This reduces the design cost for systems which employ this weighted round-robin arbitrator and speeds up the system design.

[0020] While the invention has been shown and described with reference to preferred embodiments thereof, it is well understood by those skilled in the art that various changes and modifications can be made in the invention without departing from the spirit and scope of the invention as defined by the appended claims. For example, the weighted round-robin arbitrator of the present invention can also be used for any situation where resource is shared among more than one potential users.